Brazilian Journal of Development

Maximizando a detecção de ataques com algoritmos de aprendizado de máquina através da agregação de fluxos IP

Maximizing attack detection with machine learning algorithms by aggregating IP flows

DOI:10.34117/bjdv5n10-364

Recebimento dos originais: 10/09/2019
Aceitação para publicação: 29/10/2019

Fernando Luiz Moro
Graduando em Sistemas de Informação
Instituição: Instituto Federal Catarinense – Campus Camboriú
Endereço: Rua Joaquim Garcia S/N - Caixa Postal Nº 2016 - Camboriú - SC
E-mail: fernandoluizmoro@gmail.com

Alexandre Amaral
Doutor em Engenharia Elétrica pela Universidade Estadual de Campinas
Instituição: Instituto Federal Catarinense – Campus Camboriú
Endereço: Rua Joaquim Garcia S/N - Caixa Postal Nº 2016 - Camboriú - SC
E-mail: alexandre.amaral@ifc.edu.br

Rodrigo Nogueira
Mestre em Ciência da Computação pela Universidade Federal de São Carlos
Instituição: Universidade de Coimbra
Endereço: Rua Sílvio Lima, Pólo II da Universidade de Coimbra, 3030-790 - Coimbra - PT
E-mail: rodrigonogueira@dei.uc.pt

Ana Paula Amaral
Doutora em Ciência da Computação pela Universidade Estadual de Campinas
Instituição: Instituto Federal Catarinense – Campus Camboriú
Endereço: Rua Joaquim Garcia S/N - Caixa Postal Nº 2016 - Camboriú - SC
E-mail: ana.amaral@ifc.edu.br

ABSTRACT

Machine learning has been used in cybersecurity to address the limitations of pattern identification techniques in network traffic. The existence of numerous algorithms in the literature makes the choice of which one is most suitable for the network attack detection, not be a trivial task. In this paper is performed a comparative analysis of 6 supervised machine learning algorithms evaluating the impact of the aggregation of the IP flows in the predictions, training time and test. The experiments showed that the aggregation method improves the classification and reduces the processing time of the models. In the analysis performed, the Decision Tree obtained the best balance in the results.

Key word: attack detection; stream aggregation; IP streams; machine learning; IDS

Brazilian Journal of Development

RESUMO

O aprendizado de máquina tem sido utilizado na segurança cibernética para suprir as limitações das técnicas de identificação de padrões no tráfego de rede. A existência de inúmeros algoritmos na literatura faz com que a escolha de qual é o mais adequado para a detecção de ataques, não seja uma tarefa trivial. Neste trabalho é realizada uma análise comparativa de 6 algoritmos de aprendizado de máquina supervisionado avaliando o impacto da agregação dos fluxos IP nas predições, tempo de treinamento e teste. Os experimentos mostraram que o método de agregação melhora a classificação e reduz o tempo de processamento dos modelos. Nas análises realizadas, o Decision Tree obteve o melhor equilíbrio nos resultados.

Palavras chave: detecção de ataques; agregação de fluxos; fluxos IP; aprendizado de máquina; IDS.

1. INTRODUÇÃO

Cerca de 3.138.420 GB por minuto foram transmitidos pela Internet em 2018 e estima-se que até 2020, 1,7 MB de dados serão criados por segundo para cada pessoa na terra [Ahmad 2018]. A informação gerada pelo grande volume de dados tornou-se um dos principais ativos da era atual. Consequentemente, ela se tornou alvo para as atividades criminosas que ameaçam sua disponibilidade, integridade e confidencialidade. Constantemente novos ataques são criados por indivíduos e organizações para atacar as redes de computadores com o objetivo de roubar as informações privadas [Najafabadi *et al.* 2015]. Os ataques tendem a aumentar significativamente com a *Internet of Things* (*IoT*) em que se espera mais de 80 bilhões de dispositivos interconectados até 2025 [Lobato *et al.* 2016 *apud* Clay 2015].

Inúmeros mecanismos e processos são aplicados em um ambiente de rede para fins de proteção dos dados. Um sistema de detecção de intrusão (*Intrusion Detection System* – IDS) é um mecanismo que ajuda o administrador de rede a descobrir, determinar e identificar o acesso não autorizado, duplicação, alteração e destruição de um sistema de informação [Buczak e Guven 2016]. Este tipo de mecanismo depende da coleta e processamento dos dados que podem ser obtidos através de eventos gerados em um *host* ou tráfego de rede gerado e exportado pelos dispositivos de rede. Levando em consideração a escalabilidade e a natureza da infraestrutura de rede a escolha da fonte de dados pode impactar o desempenho de um IDS. Desta maneira, uma análise baseada em fluxos que representa o resumo dos pacotes trafegados na rede, faz com que o volume de dados a serem processados seja consideravelmente menor [Kakihata *et al.* 2017].

A técnica de análise utilizada por um IDS pode influenciar na maneira como os ataques são detectados. A abordagem baseada em uso indevido (*Misuse-based*) realiza a detecção

procurando por combinações em uma base de assinaturas com ataques conhecidos, enquanto a baseada em anomalia (*Anomaly-based*), define o perfil do tráfego de rede monitorado e considera um ataque, qualquer desvio deste padrão [Hamid *et al.* 2016]. Na primeira abordagem existe a necessidade de constantes atualizações na base de dados não sendo possível identificar novos ataques. Na segunda abordagem pode ser gerada uma taxa elevada de alarmes falsos [Buczak e Guven 2016]. Os métodos de aprendizado de máquina para a detecção de ataques podem ser usados para extrair automaticamente os padrões dos dados da rede, ao contrário das técnicas baseadas em assinaturas [Najafabadi *et al.* 2015]. Assim, o desenvolvimento de uma ferramenta automatizada baseada em aprendizado de máquina pode ajudar os defensores a superarem determinadas lacunas, tornando-os mais eficazes na detecção e resposta às ameaças conhecidas e emergentes [Cisco 2018].

Diferentes técnicas de aprendizado de máquina têm sido utilizadas por aplicações ao longo dos anos, porém, nenhuma delas pode ser usada apropriadamente para todos os problemas, como apontado por [Hamid *et al.* 2016]. Inúmeros trabalhos na literatura foram propostos com o objetivo de comparar a performance dos algoritmos de aprendizado de máquina no contexto da detecção de intrusão [Hamid *et al.* 2016][Kakihata *et al.* 2017][Belouch *et al.* 2018]. Em casos como [Najafabadi *et al.* 2015], é apresentado um método de agregação dos fluxos de rede com o objetivo de gerar características mais discriminativas para melhorar o aprendizado dos modelos. Contudo, não é realizada uma comparação com os fluxos sem agregação para avaliar os benefícios do procedimento aplicado.

Propõem-se neste trabalho uma análise do impacto da agregação dos fluxos IP em 6 algoritmos de aprendizado de máquina supervisionado voltados para a detecção de ataques. Foi desenvolvido um método de agregação para reduzir o volume dos fluxos IP que serão processados com o objetivo de avaliar a influência deste procedimento na predição, tempo de treinamento e teste dos modelos. Foram criadas duas bases de dados para os experimentos a fim de aferir o real ganho do método proposto em relação aos fluxos IP não agregados. O tempo de treinamento e teste para cada classificador são apresentados com os resultados das principais métricas de avaliação utilizadas na literatura. Visando a replicabilidade dos experimentos, os melhores hiperparâmetros dos classificadores também são apresentados como uma contribuição deste trabalho.

O trabalho foi organizado de modo a demonstrar os principais procedimentos de pré-processamento adotados para a avaliação dos classificadores. A seção 2 apresenta a revisão dos trabalhos relacionados e na seção 3 são descritos os principais conceitos e algoritmos de

aprendizado de máquina selecionados. A seção 4 relata a construção da base de dados principal que foi utilizada para os experimentos. Na seção 5 o método de agregação dos fluxos IP é proposto resultando na criação de uma nova base de dados utilizada para as comparações. A modelagem dos classificadores e a divisão da base de dados em conjuntos específicos são apresentadas na seção 6. Na seção 7, os resultados da análise são relatados e o melhor classificador para o cenário de detecção de ataques é identificado.

2. TRABALHOS RELACIONADOS

[Najafabadi *et al.* 2015] propõem uma abordagem de aprendizado de máquina para a detecção de ataques de força bruta. Foram coletados da rede do campus dos autores, ataques de força bruta real e falhas de login com o objetivo de simular tentativas de acesso de usuários legítimos que se esqueceram de suas senhas. Os autores identificaram que as características dos fluxos entre uma falha de login e um ataque de força bruta não eram discriminativas o suficiente. Para contornar essa situação, foi desenvolvido um método para agregar os fluxos com o mesmo IP de origem, IP de destino e porta de destino observados durante um intervalo de 5 minutos. A base de dados gerada contém 425 instâncias anômalas, 558 normais e um total de 19 características. Os algoritmos escolhidos foram o 5-*Nearest Neighbor*, C4.5D, C4.5N e *Naive Bayes*, ambos provenientes da ferramenta Weka. Foi executado 4 vezes o 5-*folds cross-validation* em conjunto com *Area Under the receiver operating characteristic Curve* (AUC). Foi calculado a média final com base em 20 AUC e os resultados demonstram uma performance acima de 98% para todos os classificadores. Os autores concluem que a boa performance dos modelos foi devido a agregação realizada que resultou em características mais discriminativas para diferenciar um ataque de força bruta.

[Hamid *et al.* 2016] realizam uma análise comparativa de diferentes técnicas de aprendizado de máquina supervisionado da ferramenta Weka para a detecção de ataques. Foi utilizado 10% da base de dados KDD CUP 1999, totalizando 494.020 amostras de dados brutos TCP simulados, compostos por 22 tipos de ataques divididos em cinco principais grupos. Os autores selecionaram as métricas de avaliação oriundas da matriz de confusão e aplicaram a técnica de 10-*folds cross-validation* levando em consideração a média dos resultados gerados. Foi relatado que o classificador *PART* do grupo *Rule Based* obteve a melhor performance de 99,97% e o *Input Mapped* teve o pior resultado, com um total de 56,83%. Foi levado em consideração a técnica de redução de características com o objetivo de diminuir a complexidade do conjunto de dados. Foram aplicadas diversas técnicas e foi selecionado o

avaliador *CfsSubSetEval* com o método de busca *BestFirst*. O resultado foi a seleção de 11 características das 41, que possibilitaram manter os resultados semelhantes aos obtidos anteriormente reduzindo o custo computacional.

[Kakihata *et al.* 2017] apresentam um IDS que verifica o comportamento da rede por meio de consultas a um banco de dados e posteriormente compara os fluxos com três assinaturas com o objetivo de informar um ataque. Os autores relatam que por meio da análise e comparação foi possível detectar de forma eficiente todos os ataques da base de dados. Foi desenvolvido uma aplicação em Java para rotular os fluxos previamente detectados para serem usados pelos algoritmos *Optimum-Path Forest* (OPF), *Support Vector Machine* (SVM), *K-Nearest Neighbors* (KNN) e *Bayesian*. Para a realização do treinamento dos algoritmos de aprendizado de máquina, foi considerado uma base de dados com o total de 10.000 fluxos normais e 11.413 fluxos anômalos sendo apenas 10% deste montante, destinado para a predição. Os classificadores que obtiveram as melhores acurácias foram o OPF com 99,62% e o *Bayesian* com 96,62%. Os autores realizaram um segundo teste utilizando os dados normalizados destacando-se o classificador *Bayesian* com 99,62% e o KNN com 99,60% de acurácia. Foram avaliados o tempo de processamento dos modelos podendo concluir neste quesito, que o classificador *Bayesian* possui o menor tempo para o treinamento e o SVM para a etapa de predição.

[Belouch *et al.* 2018] avaliam a performance de quatro algoritmos de aprendizado de máquina bem conhecidos como o *SVM*, *Naive Bayes*, *Decision Tree* e *Random Forest* utilizando o Apache Spark. Foi utilizada uma base de dados recente denominada UNSW-NB15 contendo tráfego de rede normal e anômalo. Com cerca de 49 características e 9 famílias de ataques, a base de dados possui um conjunto de treinamento com 175.340 amostras e um de teste contendo 82.000 amostras. Para avaliar o desempenho dos algoritmos foram utilizadas as métricas de acurácia, sensibilidade, especificidade, tempo de treinamento e tempo de teste. Os autores salientam que o *Random Forest* obteve o melhor desempenho no tempo de teste totalizando 0,08 segundos com uma acurácia de 97,49%. O pior desempenho dos classificadores foi o *Naive Bayes* que obteve uma acurácia de 74,19%, embora tenha atingido o menor tempo de treinamento com um total de 2,25 segundos.

Ambos os trabalhos revisados apresentam uma análise comparando os algoritmos de aprendizado de máquina para a detecção de ataques. A contribuição apresentada neste trabalho e que se distingue dos demais, consiste na comparação dos algoritmos de aprendizado de máquina com base nos fluxos IP com e sem agregação. O objetivo é avaliar a influência que o

método de agregação causa no tempo de processamento e nos resultados das predições. As características utilizadas para a agregação e a janela temporal para a coleta dos fluxos diferem-se de [Najafabadi *et al.* 2015] que embora apresentem um método para agregar os fluxos IP, não comparam os resultados em relação aos fluxos não agregados. Desta maneira, não é possível aferir o real impacto no tempo de processamento dos algoritmos. A utilização de uma janela temporal menor permite detectar antecipadamente um ataque, pois aguardar um longo período pode resultar em sérios prejuízos para a rede [Moro *et al.* 2018]. Outra contribuição deste trabalho em relação aos apresentados na literatura é a busca e retorno dos melhores hiperparâmetros para cada classificador.

3. ALGORITMOS DE APRENDIZADO DE MÁQUINA

O aprendizado de máquina é um subcampo da inteligência artificial que dá ao computador a capacidade de aprender sem ser explicitamente programado [Das e Nene 2017]. Dependendo dos objetivos e da natureza da base de dados podem ser utilizada três abordagens principais conhecidas como aprendizado supervisionado, não supervisionado e por reforço. No aprendizado supervisionado com foco na classificação, cada dado é mapeado para uma das classes que se deseja predizer. Mediante a uma nova entrada, o modelo encontrará com base nos padrões aprendidos a classe a qual o dado pertence. Em problemas de clusterização relacionados ao aprendizado não supervisionado, os dados não possuem rótulos e o objetivo é descobrir grupos baseados em suas similaridades ou dissimilaridades [Scikit-learn 2018]. O aprendizado por reforço utiliza um agente responsável por tomar decisões com base na tentativa e erro sendo recompensado caso haja sucesso, assim, gradativamente o aprendizado é desenvolvido [AltexSoft 2018].

No contexto de detecção de ataques, o uso de aprendizado de máquina tem aumentado devido às limitações das técnicas tradicionais (*e.g.*, *firewall*, controle de acesso) em detectar ataques mais sofisticados [Najafabadi *et al.* 2015 *apud* Zamani e Movahedi 2013]. Com o enfoque em detectar ataques de rede por meio de uma análise do comportamento dos fluxos IP e devido a base de dados utilizada nos experimentos estar devidamente rotulada, optou-se pelo aprendizado de máquina supervisionado voltado para a técnica de classificação. Com base nas pesquisas realizadas por [Buczak e Guven 2016] e [Das e Nene 2017] sobre as técnicas de aprendizado de máquina para a segurança cibernética, foram escolhidos os classificadores *Decision Tree*, *Random Forest, Gaussian Naive Bayes*, *K-Nearest Neighbors*, *Support Vector Machine* e *Multi-layer Perceptron*. A implementação destes algoritmos e os

demais procedimentos mencionados no decorrer do texto foram realizados através do módulo scikit-learn da linguagem de programação Python. A Tabela 1 apresenta uma breve descrição dos classificadores escolhidos para os experimentos com base em [Buczak e Guven 2016][Lobato *et al.* 2016][Najafabadi *et al.* 2015][Scikit-learn 2018]:

Tabela 1. Algoritmos de aprendizado de máquina supervisionado.

Algoritmos	Descrição
Decision Tree	Estrutura de árvore composta por folhas que representam as classes e ramos compostos por regras de decisão baseadas nas características.
Random Forest	Aprendizado em conjunto (*ensemble learning*) que gera uma coleção de árvores de decisão independentes combinando os resultados para formar uma melhor hipótese.
Gaussian Naive Bayes	Teorema de Bayes que supõe de forma ingênua a independência condicional entre cada par de características dado o valor da classe. A probabilidade das características é assumida como Gaussiana.
K-Nearest Neighbors	Aprendizado em instância (*instance learning*) que utiliza k amostras de treinamento mais próximas em distância de uma nova entrada para atribuir uma classe a ela.
Support Vector Machine	Descoberta de um hiperplano de separação em um espaço multidimensional que divide diferentes classes maximizando a margem entre elas.
Multi-Layer Perceptron	Rede de neurônios artificiais baseada no cérebro humano que processa os dados de entrada transferindo o resultado entre as camadas até chegar na última onde a classe é definida.

4 BASE DE DADOS

Um dos principais pontos para a obtenção de bons resultados no aprendizado de máquina

é uma base de dados com amostras representativas do problema que se pretende resolver. No entanto, as bases de dados utilizadas amplamente na literatura, tais como a DARPA 1998, DARPA 1999 e o KDD 1999 possuem tráfego sintetizado e com grande número de redundâncias causando viés e consequentemente afetando o resultado dos algoritmos [Buczak e Guven 2016]. Alternativamente, o NSL-KDD foi criado com o objetivo de resolver os problemas apresentados pelos seus antecessores, porém, o tráfego de rede normal e os ataques gerados não representam mais o cenário atual das redes de computadores [Belouch *et al.* 2018 *apud* Tavallaee *et al.* 2009].

Para contornar esta questão, considerou-se neste trabalho uma base de dados com 95 GB de pacotes capturados do laboratório do Grupo de Teleinformática e Automação (GTA) da Universidade Federal do Rio de Janeiro (UFRJ), contendo tráfego normal e ataques de rede reais divididos em duas principais classes: *Denial of Service* (DoS) e *Probe* [Lobato *et al.* 2016]. Os ataques que compõem a primeira classe são, *ICMP flood*, *land*, *nestea*, *smurf*, *SYN flood*, *teardrop* e *UDP flood*. Nos ataques destinados a varredura de rede e sistemas foram utilizados o *TCP SYN scan*, *TCP connect scan*, *SCTP INIT scan*, *Null scan*, *FIN scan*, *Xmas scan*, *TCP ACK scan*, *TCP Window scan* e *TCP Maimon scan*.

A base de dados é um dos fatores que determinam quais os tipos de ataques serão detectados, implicando também na escalabilidade, que deve ser levada em consideração para a implantação da solução em grandes redes [Amaral *et al.* 2017]. Portanto, todos os pacotes com comportamento normal foram convertidos para fluxos IP com base no padrão Netflow através da ferramenta Nfpcapd considerando uma janela temporal de 60 segundos. Os pacotes contendo os ataques foram convertidos para produzir a mesma quantidade que a anterior devido a proporção de ataques na base de dados ser maior em relação ao tráfego normal. Posteriormente, por meio do Nfdump foi possível processar estes dados e transformá-los em fluxos IP minimizando assim, o volume de dados coletados, processados e armazenados [Moro et al. 2018]. Um fluxo é uma sequência de pacotes unidirecional com propriedades em comum como IP de origem e destino, porta de origem e destino e protocolo observados em um intervalo de tempo [IETF 2018].

Optou-se pela construção de uma base de dados balanceada para evitar que os algoritmos de aprendizado de máquina fossem tendenciosos em relação a uma classe específica. Cada amostra da base de dados foi rotulada como normal (0) ou anômala (1) dependendo das suas características. Conforme o procedimento mencionado acima foram gerados um total de 68.872 fluxos IP contendo comportamento normal dos quais, uma amostra de 50.000 foi

coletada. Semelhantemente, 50.000 fluxos IP anômalos foram coletados contendo todos os tipos de ataques supracitados. Assim, a base de dados resultou em um total de 100.000 fluxos IP com 48 características armazenadas em um arquivo CSV (*Comma-separated values*).

5 PRÉ-PROCESSAMENTO

O pré-processamento dos dados é um aspecto importante para alcançar o conjunto de dados final que será utilizado durante a criação dos modelos [Buczak e Guven 2016]. Inicialmente foi observado que muitas características da base de dados não contribuiriam para o aprendizado por não serem representativas ou por conterem valores nulos. Assim, foram selecionadas 3 características principais de um fluxo IP: duração em segundos (*td*), quantidade de pacotes (*pkt*) e quantidade de bytes (*byt*). Visando acrescentar novos metadados, foram criadas 3 novas características a partir das existentes denominadas bits por segundos (*bps*), bits por pacotes (*bpp*) e pacotes por segundo (*pps*). A base de dados final resultou em um total de 6 características conforme exemplo apresentado na Tabela 2 em negrito. Os IPs de origem e destino foram omitidos por serem endereços públicos.

Tabela 2. Fluxos IP com as características selecionadas em negrito.

sa	da	pr	flg	sp	dp	**td**	**pkt**	**byt**	**bps**	**bpp**	**pps**	flw
.....165	173	udp	[0]	58853	547	**5**	**5**	**3648**	**5837**	**5837**	**1**	1
.....165	119	tcp	[0, 1, 1, 0, 0, 1]	54547	443	**8**	**65**	**5420**	**5420**	**667**	**8**	1
.....165	119	tcp	[0, 1, 1, 0, 1, 1]	54552	443	**13**	**12**	**900**	**554**	**600**	**1**	1
.....165	119	tcp	[0, 1, 1, 0, 1, 1]	54551	443	**13**	**11**	**868**	**534**	**631**	**1**	1

Durante o procedimento de pré-processamento, foi analisado se a agregação dos fluxos IP impactaria no tempo de processamento e resultados dos classificadores. Desta maneira, foi desenvolvido um método para agregar os fluxos IP com base no IP de origem, IP de destino e protocolo em um intervalo de 60 segundos. A Tabela 3 apresenta o resultado da aplicação deste método nos fluxos descritos na Tabela 2.

Tabela 3. Fluxos IP agregados com as características selecionadas em negrito.

sa	da	pr	flg	**qsp**	**qdp**	**td**	**pkt**	**byt**	**bps**	**bpp**	**pps**	**flw**
.....165	173	udp	[0]	**1**	**1**	**5**	**5**	**3648**	**5837**	**5837**	**1**	**1**
.....165	119	tcp	[0, 3, 3, 0, 2, 3]	**3**	**1**	**13**	**88**	**7188**	**4423**	**653**	**7**	**3**

Brazilian Journal of Development

Nota-se na Tabela 2 que as características de porta de origem (*sp*) e porta de destino (*dp*) não foram utilizadas, pois o valor não era representativo para o aprendizado. Através do método de agregação foi possível transformar essas características em quantidade de porta de origem (*qsp*) e quantidade de porta de destino (*qdp*), não sendo contabilizadas as portas duplicadas. A agregação das portas é importante em ataques que geram um alto volume de fluxos (*e.g., flooding*) e utilizam as portas de origem aleatória. Na Tabela 2 os três últimos fluxos duraram respectivamente 8, 13 e 13 segundos, contudo, ambos iniciaram no mesmo período (13:41:08) e finalizam em tempos diferentes (13:41:16, 13:41:21 e 13:41:21). Assim, foi considerado somente o término da comunicação representado pelo período de maior valor. Os valores para *bps*, *bpp*, *pps* foram recalculados com base nos novos valores agregados e a quantidade de fluxos na comunicação (*flw*) foi considerada por permitir distinguir ataques DoS específicos do tráfego normal.

Através do método proposto foi possível criar uma nova base de dados para ser utilizada em conjunto com a anterior permitindo a comparação da influência da agregação nos resultados apresentados pelos modelos. Foi estabelecido um limite no método de agregação visando manter a base de dados agregada balanceada devido à existência de casos em que milhares de fluxos seriam reduzidos para uma única amostra. Foi definido que durante um intervalo de 60 segundos, se a quantidade de fluxos (*flw*) atingir a quantidade de 100, a agregação é interrompida e um novo fluxo é iniciado. A quantidade de fluxos final em cada uma das bases de dados é apresentada na Figura 1.

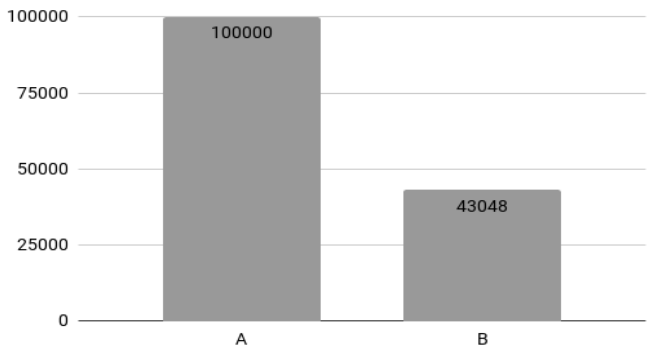

Figura 1. Quantidade de fluxos nas bases de dados

A aplicação do método de agregação na base de dados B reduziu 56,95% a quantidade de fluxos que os modelos deverão processar em relação a base de dados A.

Brazilian Journal of Development

6 MODELAGEM

A modelagem consiste em uma etapa onde os algoritmos de aprendizado de máquina são treinados e os hiperparâmetros são otimizados [Buczak e Guven 2016]. Foram adotados alguns procedimentos comumente encontrados na literatura para a aplicação em cada uma das bases de dados. Primeiramente, os dados foram divididos aleatoriamente em um conjunto de treinamento contendo 70% das amostras e outro conjunto destinado para o teste com os 30% restantes. Esta abordagem é importante, pois avaliar a habilidade de um classificador nos conjuntos de treinamento resultaria em uma pontuação tendenciosa, assim, para dar uma estimativa imparcial sobre a habilidade do modelo o conjunto de teste é utilizado [Brownlee 2017]. Durante a divisão dos conjuntos de dados foi aplicada a técnica de estratificação, que preserva em cada parte a mesma proporção de amostras de cada classe em relação a base de dados original [Scikit-learn 2018].

Os algoritmos de aprendizado de máquina possuem um conjunto de hiperparâmetros, que são parâmetros não aprendidos diretamente durante o treinamento e que podem impactar na predição e performance computacional dos modelos [Scikit-learn 2018]. Deste modo, foi utilizado o *Grid Search* em conjunto com *Stratified* 5-*fold cross-validation* com o objetivo de buscar a melhor combinação de hiperparâmetros para cada classificador. A técnica de *cross-validation* é aplicada no conjunto de treinamento realizando a divisão de acordo com o parâmetro k estabelecido, sendo usado neste trabalho k=5. Semelhante a divisão inicial, este procedimento separa um conjunto isolado para avaliar a habilidade do modelo preservando a proporção de cada classe durante as divisões. Através do *Grid Search* são geradas inúmeras combinações de um mesmo modelo de acordo com os hiperparâmetros definidos. Cada combinação será treinada em k-1 divisões e avaliada em um conjunto de validação. Este procedimento é aplicado por 5 vezes seguidas alternando a ordem das divisões utilizadas. Ao finalizar o *cross-validation* uma combinação foi avaliada e a média dos resultados obtidos é armazenada. Após o término, o *cross-validation* inicia novamente para uma outra combinação e o melhor resultado é escolhido como solução. A Figura 2 ilustra as divisões aplicadas nas bases de dados pelas diferentes técnicas adotadas.

Brazilian Journal of Development

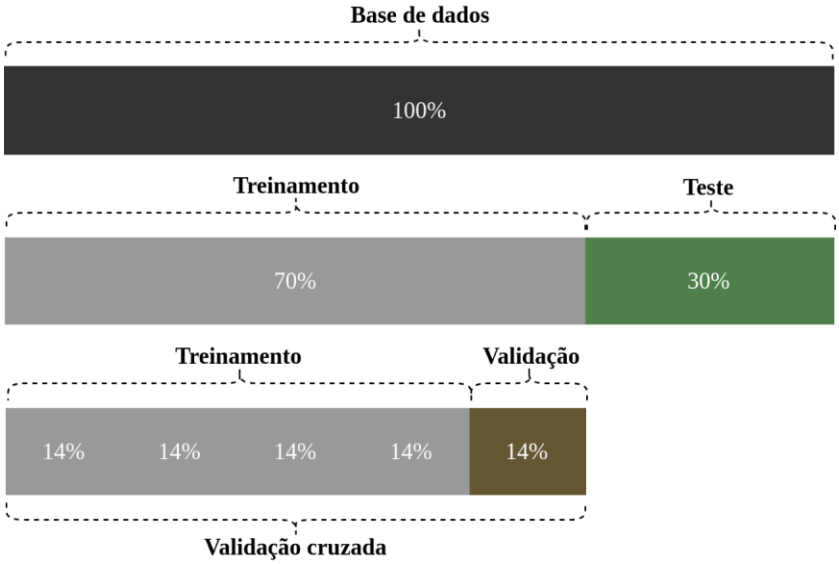

Figura 2. Divisões aplicadas nas bases de dados

Vale ressaltar que os valores de 5 e 10 para o parâmetro *k* do *cross-validation* são comumente utilizadas na literatura [Hamid *et al.* 2016][Lobato *et al.* 2016][Najafabadi *et al.* 2015]. Neste trabalho optou-se pela utilização de *k*=5, pois foram encontrados resultados semelhantes através de testes realizados na base de dados A com *k*=10, contudo, houve um aumento de 55,84% no tempo de treinamento. Com os classificadores treinados e os hiperparâmetros selecionados é realizado a predição para o conjunto de teste que representam os dados não observados durante o treinamento. O resultado da predição que consiste nas classes estimadas por um modelo é utilizado como base para as métricas de avaliação.

7 AVALIAÇÃO

Para avaliar os algoritmos de aprendizado de máquina supervisionado foram considerados o tempo de treinamento e teste, assim como, as métricas de acurácia, precisão, revocação e *f1-score* originadas da matriz de confusão definidas em [Buczak e Guven 2016][Hamid *et al.* 2016][Belouch *et al.* 2018]. Para a realização da análise foi conduzido um experimento para a base de dados A e outro para B envolvendo os classificadores selecionados. Foi utilizado uma máquina virtual com sistema operacional Debian GNU/Linux 9.4 em um servidor com processador Intel Xeon CPU E5-2640 com 8 núcleos de 2.50 GHz, 24 GB de memória RAM e 600 GB de disco rígido. Os resultados para o primeiro experimento são apresentados na Tabela 4.

Brazilian Journal of Development

Tabela 4. Análise comparativa dos classificadores na base de dados A.

Algoritmos	Acurácia	Precisão	Revocação	*F1-score*	Tempo de treinamento (s)	Tempo de teste (s)
Decision Tree	99,15%	98,37%	99,95%	99,16%	319,5663	0,0335
Random Forest	99,17%	98,38%	99,99%	99,18%	1042,6025	0,0556
Gaussian Naive Bayes	59,47%	55,23%	100,00%	71,16%	4,7906	0,0359
K-Nearest Neighbors	99,12%	98,37%	99,91%	99,13%	1514,2970	3,6116
Support Vector Machine	99,10%	98,42%	99,81%	99,11%	1884,6281	7,5794
Multi-Layer Perceptron	90,48%	97,73%	82,89%	89,70%	31326,8961	0,0469

Conforme os resultados descritos na Tabela 4 todos os classificadores alcançaram uma acurácia acima de 90%, com exceção do *Gaussian Naive Bayes*. De modo geral, todas as métricas utilizadas obtiveram resultados satisfatórios demonstrando a capacidade de aprendizado dos modelos. Observando os menores tempos de treinamento o *Gaussian Naive Bayes* destacou-se com 4,7906 segundos sendo seguido pelo *Decision Tree* com 319,5663 segundos. O algoritmo que consumiu o maior tempo durante o aprendizado e a busca pelos hiperparâmetros foi o *Multi-Layer Perceptron* por conta da sua complexidade computacional. O *Support Vector Machine* demorou cerca de 7,5794 segundos para classificar as amostras no conjunto de teste, enquanto que o *Decision Tree* levou apenas 0,0335 segundos. Para esta primeira análise os classificadores *Decision Tree* e *Random Forest* obtiveram os melhores resultados, diferindo-se no tempo de processamento em que o *Decision Tree* obteve vantagem devida a criação de apenas uma estrutura de árvore.

Vale ressaltar que o tempo de treinamento e teste consideraram as etapas de aprendizado e busca dos hiperparâmetros. No entanto, cada classificador possui os próprios hiperparâmetros. Como exemplo, nos experimentos foram definidos 4 para o *K-Nearest Neighbors* e apenas 1 para o *Gaussian Naive Bayes,* cada qual, com múltiplas opções (*e.g., n_neighbors*: [5, 10, 15]). Além da complexidade computacional de cada modelo, as inúmeras combinações geradas pelo *Grid Search* influenciam diretamente no tempo de treinamento. Embora a comparação deste quesito não seja tão significativa entre os algoritmos da base de dados A, ao compará-los em relação aos tempos de treinamento e teste da base de dados B é possível avaliar o impacto da agregação dos fluxos IP. A Tabela 5 apresenta os resultados para a base de dados B.

Brazilian Journal of Development

Tabela 5. Análise comparativa dos classificadores na base de dados B.

Algoritmos	Acurácia	Precisão	Revocação	*F1-score*	Tempo de treinamento (s)	Tempo de teste (s)
Decision Tree	99,99%	100,00%	99,99%	99,99%	164,7494	0,0174
Random Forest	99,99%	100,00%	99,97%	99,99%	578,1899	0,0276
Gaussian Naive Bayes	63,24%	57,63%	100,00%	73,12%	2,5123	0,0199
K-Nearest Neighbors	99,61%	99,51%	99,72%	99,61%	295,7518	0,6677
Support Vector Machine	97,24%	100,00%	94,47%	97,16%	982,5447	3,4181
Multi-Layer Perceptron	99,67%	99,88%	99,46%	99,67%	11807,2571	0,0455

Note-se que houve um aumento na taxa de precisão de todos os algoritmos. Com base nos fluxos IP agregados os classificadores baseados em estrutura de árvore e o *Support Vector Machine* conseguiram detectar e classificar todas as amostras anômalas analisadas no conjunto de teste. Em relação aos resultados da Tabela 4 os algoritmos apresentaram uma maior taxa de acerto, com exceção do *Support Vector Machine* que teve sua acurácia reduzida em 1,86%. Destaca-se o *Multi-Layer Perceptron* que aumentou consideravelmente a acurácia em 9,19%. O tempo de treinamento total consumido pelos modelos na base de dados A foi de 36092,7806 segundos. Na base de dados B, este valor foi para 13831,0052 segundos representando uma redução de 61,68%. De modo semelhante, o tempo de teste reduziu 63,07%. Estes dados demonstram que o método de agregação permitiu aumentar a precisão e reduzir consideravelmente o tempo de processamento dos classificadores.

Em linhas gerais, o algoritmo que se mostrou mais adequado para o cenário de detecção de ataques, apresentando um equilíbrio entre as métricas de avaliação e o tempo de processamento foi o *Decision Tree*. Por meio do método de agregação desenvolvido o tempo de treinamento deste algoritmo diminui 48,45% em comparação com a base de dados A, e o tempo de teste reduziu 48,06%. Além disso, houve um aumentou de 0,84% na acurácia atingindo 99,99%. O *Gaussian Naive Bayes* apresentou em ambos os experimentos resultados insatisfatórios, mesmo sendo considerado o algoritmo mais rápido. Os melhores hiperparâmetros de cada algoritmo de aprendizado de máquina supervisionado são apresentados na Tabela 6.

Brazilian Journal of Development

Tabela 6. Hiperparâmetros encontrados para as bases de dados A e B.

Algoritmos	Hiperparâmetros - Base de dados A	Hiperparâmetros - Base de dados B
Decision Tree	criterion: gini, max_depth: 15, max_features: sqrt, min_samples_leaf: 2, min_samples_split: 10, splitter: best	criterion: entropy, max_depth: 9, max_features: sqrt, min_samples_leaf: 2, min_samples_split: 10, splitter: best
Random Forest	criterion: gini, max_depth: 15, max_features: sqrt, min_samples_leaf: 2, min_samples_split: 5, n_estimators: 10	criterion: entropy, max_depth: 15, max_features: sqrt, min_samples_leaf: 5, min_samples_split: 10, n_estimators: 10
Gaussian Naive Bayes	var_smoothing: 1e-09	var_smoothing: 1e-09
K-Nearest Neighbors	algorithm: ball_tree, leaf_size: 10, n_neighbors: 15, weights: distance	algorithm: ball_tree, leaf_size: 10, n_neighbors: 5, weights: distance
Support Vector Machine	C: 10.0, cache_size: 5000.0, gamma: 0.01, kernel: rbf	C: 10.0, cache_size: 5000.0, gamma: 0.001, kernel: rbf
Multi-Layer Perceptron	activation: relu, alpha: 0.1, hidden_layer_sizes: (20, 15, 10), max_iter: 500, solver: adam	activation: relu, alpha: 0.0001, hidden_layer_sizes: (15, 10), max_iter: 100, solver: adam

O objetivo de apresentar os dados da Tabela 6 consiste em permitir a replicação dos experimentos realizados neste trabalho. Para a melhor compreensão do funcionamento dos hiperparâmetros consulte a documentação em [Scikit-learn 2018].

8 CONSIDERAÇÕES FINAIS

Foi proposto neste trabalho a agregação dos fluxos IP com objetivo de realizar uma análise do impacto deste procedimento em 6 algoritmos de aprendizado de máquina supervisionado aplicados para a detecção de ataques. Duas bases de dados foram criadas para os experimentos considerando os fluxos IP com e sem agregação. Em linhas gerais através dos experimentos e análises realizadas os classificadores obtiveram bons resultados em ambas as bases de dados. O *Decision Tree* apresentou uma acurácia de 99,15% em um cenário com dados não agregados e 99,99% utilizando a base de dados com agregação. Este algoritmo também apresentou um equilíbrio entre o tempo de treinamento e teste, sendo considerado o mais adequado para a detecção de ataques nos cenários avaliados. O *Gaussian Naives Bayes* apresentou a pior acurácia entre os demais classificadores e o *Multi-Layer Perceptron* atingiu um aumento de 9,19% na acurácia com os fluxos IP agregados. O método de agregação

permitiu aumentar a taxa de precisão dos algoritmos e diminuir o volume de dados a serem processados. Através dele foi possível reduzir em 61,68% o tempo de treinamento e 63,07% o tempo de teste. Os melhores hiperparâmetros encontrados para cada modelo foram apresentados para permitir a replicabilidade dos experimentos. Como trabalho futuro será desenvolvido uma solução para detecção de ataques utilizando aprendizado de máquina para atuação em tempo real incorporando ações de mitigação.

REFERÊNCIAS

Ahmad, I. (2018). How Much Data Is Generated Per Minute? The Answer Will Blow Your Mind Away. https://www.digitalinformationworld.com/2018/06/infographics-data-never-sleeps-6.html, accessed on November.

AltexSoft (2018). Machine Learning: Bridging Between Business and Data Science. . https://www.altexsoft.com/whitepapers/machine-learning-bridging-between-business-and-data-science/, accessed on November.

Amaral, A. A., Mendes, L. de S., Zarpelão, B. B. and Junior, M. L. P. (2017). Deep IP flow inspection to detect beyond network anomalies. *Computer Communications*, v. 98, p. 80–96.

Belouch, M., El Hadaj, S. and Idhammad, M. (2018). Performance evaluation of intrusion detection based on machine learning using Apache Spark. *Procedia Computer Science*, v. 127, p. 1–6.

Brownlee, J. (2017). What is the Difference Between Test and Validation Datasets? *Machine Learning Mastery*. https://machinelearningmastery.com/difference-test-validation-datasets/, accessed on Nov.

Buczak, A. L. and Guven, E. (2016). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys & Tutorials*, v. 18, n. 2, p. 1153–1176.

Cisco (2018). Cisco 2018 Annual Cybersecurity Report. https://www.cisco.com/c/en/us/products/security/security-reports.html, accessed on November.

Das, S. and Nene, M. J. (2017). A survey on types of machine learning techniques in intrusion prevention systems. In *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. IEEE.

http://ieeexplore.ieee.org/document/8300169/, accessed on November.

Hamid, Y., Sugumaran, M. and Journaux, L. (2016). Machine Learning Techniques for Intrusion Detection: A Comparative Analysis. In *Proceedings of the International Conference on Informatics and Analytics – ICIA-16*. ACM Press. http://dl.acm.org/citation.cfm?doid=2980258.2980378, accessed on October.

IETF (2018). IP Flow Information Export (IPFIX). http://datatracker.ietf.org/wg/ipfix/charter/, accessed on Nov.

Lobato, A. G. P., Lopez, M. A. and Duarte, O. C. M. B. (2016). An Accurate Threat Detection System through Real-Time Stream Processing. Grupo de Teleinformática e Automação (GTA), Universidade Federal do Rio de Janeiro (UFRJ), Tech. Rep.

Kakihata, E. M., Sapia, H. M., Oiakawa, R. T., et al. (2017). Intrusion Detection System Based On Flows Using Machine Learning Algorithms. *IEEE Latin America Transactions*, v. 15, n. 10, p. 1988–1993.

Moro, F. L., Amaral, A. A., Amaral, A. P. M. and Nogueira, R. R. (2018). Detecção e mitigação de um ataque DoS em seu estágio inicial em uma rede definida por software. In *IX Congresso Sul Brasileiro de Computação (SULCOMP).*

Najafabadi, M. M., Khoshgoftaar, T. M., Calvert, C. and Kemp, C. (2015). Detection of SSH Brute Force Attacks Using Aggregated Netflow Data. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE. http://ieeexplore.ieee.org/document/7424322/, accessed on October.

Scikit-learn (2018). Supervised learning – scikit-learn 0.20.0 documentation. https://scikit-learn.org/stable/supervised_learning.html, accessed on November.